MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 836 April, 1985
revised June, 1985

Deterministic Parsing and Linguistic Explanation

Robert C. Berwick
MIT Artificial Intelligence Laboratory

Amy S. Weinberg
University of Maryland, College Park

ABSTRACT:

This article summarizes and extends recent results linking deterministic parsing to observed "locality principles" in syntax. It also argues that grammatical theories based on explicit phrase structure rules are unlikely to provide comparable explanations of why natural languages are built the way they are.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research has been provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505. The authors are indebted to Noam Chomsky, Janet Fodor, Norbert Hornstein, Boris Katz, and Ann Riley for assistance in writing this paper.

1 Introduction

A cognitive approach to language asks both representational and computational questions. Our aim in our recent work, summarized in *The Grammatical Basis of Linguistic Performance*— is to discover both *what* our knowledge of language is—a question about representation— and *how* that knowledge is put to use—a question about computation. We argued—and we'll reinforce that argument here—that we can gain a deeper understanding of why natural languages are built the way they are by considering how the problems of efficient parsing and learning connect to the representation of grammars. We showed that *if* one is willing to make a few strong but natural assumptions about constraints on human parsing abilities and how grammars are used as parsers, then one can show, in part, why locality constraints like Subjacency must be a part of grammatical descriptions. Our assumptions were these:

- Parsing is *deterministic*, in the sense that once information about the structure of a sentence is written down, it is never retracted. This means that the information about a sentence is monotonically preserved during analysis.
- Grammatical representations are embedded *directly* into parsers, without intervening derived predicates or multiplied-out rule systems. This is an assumption of *transparency* (Berwick and Weinberg 1984).
- The human brain is finite.

The assumptions about determinism and transparency are strong, but, as we'll see, natural. They are meant to be. Our explanatory punch works in direct proportion to the strength of the constraints: if we adopt a system where anything goes, then we cannot explain why languages are built one way rather than another.

Naturally—and fortunately— this leaves the system of assumptions open to refutation. In a recent article to appear in *Language and Cognitive Processes* (1985), Janet Fodor takes issue with both the linguistic details behind the theory of grammar we adopt and with the assumptions of monotonicity and transparency. We believe that each of these criticisms falls short, and we'll survey just what Fodor says as well as our own position, but before launching into a bill of particulars, it's worthwhile to step back and survey the approach Fodor implicitly endorses.

There's a style of theory construction in A.I. that might be dubbed "universal simulation." The idea is to adopt the *weakest* possible set of assumptions about a computational process, for fear of being wrong. A lampoon version goes something like this: (i) every cognitive process is a computational process; (ii) Turing machines can simulate any computational process; so (iii) I'd better adopt a Turing machine as a model of this cognitive process, because

otherwise I may miss something. That's sheer hyperbole, of course, but something disturbingly close to this lies behind the embrace of nondeterminism as a central feature of parsing models. The problem, as we specifically observe in our book and as Fodor echoes, is that since nondeterministic computation subsumes deterministic computation, one can always simulate the effect of the deterministic assumption simply by making the cost of nondeterminism very high. What Fodor fails to note is the flip side to this point: one can always get the functional effect of recovery from failed determinism, such as garden paths, by adding recovery procedures to deterministic parsers. So why all the fuss? Don't these two apparently opposed camps just merge into a gray middle ground?

The difference is one of point of view and methodological stance. Forcing an essentially nondeterministic procedure to be deterministic by adding cost to backup violates the spirit of nondeterministic computation precisely in the same way that arbitrary backtracking would violate the spirit of determinism. We prefer to make the stronger--and more refutable -hypotheses about transparency and determinism. We'd argue that recovery from garden paths and near garden paths need not cause a deterministic parser to throw up its hands, but invokes quite particular, non-ad hoc reconstruction procedures that use the information built up about the parse *in a deterministic way*. More about that later. The important point here is that we adopt the determinism requirement as a basic article –a "leading idea," to be weakened only under duress and in quite limited, particular cases. In contrast, based on the same evidence, Fodor adopts nondeterminism as a leading idea. These different positions lead to quite different ways of thinking about parsing. For someone who endorses nondeterminism, the hard part isn't figuring out how parsing gets done –that's *easier*, because we have more machinery at our disposal the hard part is figuring out what the constraints are and how to naturally enforce them. We must now be able to say why parsing isn't done some other way that is just as easy to encode using the extra machinery of nondeterminism. Plainly the burden of proof here falls on Fodor's shoulders; her position is the weaker one. One example of this point should suffice. Fodor argues that adding an extra memory cell or its functional equivalent to a transition network parser (e.g., a hold cell) makes parsing easy. Therefore, she concludes, it should be added. More strikingly, she comments: "B[erwick] and W[einberg] simply have to **stipulate** that their parser has no such facility." (page 50; our emphasis). But since when does one have to stipulate the *nonexistence* of additional machinery? As Marcus (1980:146) says on this point, "What demands explanation and motivation is why a given facility *is* included in the model .... Thus, there is no reason to explain why a mechanism of only limited power has been implemented if it can be shown that it is enough to the job that is required." What is more, by sticking to more restricted machinery, we can actually explain some of the structural characteristics of natural languages.

Of course our leading idea may be incorrect. Then we will be led, regrettably, to nondeterminism, to nontransparency, and perhaps beyond. We say regrettably, because then we will be in a weaker position. Once the Pandora's box of unlimited nondeterministic computation is opened, we can nail it shut only by importing constraints from other domains. Again, this may be possible; we cannot rule it out. Fodor hints at constraints on grammar size having to do with parsing/learnability – but we'll see these arguments lack support. Simply put, the search space of nondeterministically- and nontransparently-based theories is much vaster. We prefer to start with the much smaller world of determinism and work outwards.

We were well aware of this difficulty in our book. That's why we took great pains to distinguish between *two* versions of nondeterminism: (1) "true" nondeterminism in parsing, where all interpretations are carried along simultaneously; and (2) "backtracking" nondeterminism, where all nondeterministic alternatives are explored one at a time. We carefully observed that our functional arguments bifurcating deterministic and nondeterministic parsing applied only to true nondeterminism. By thinking about this contrast, we were led to quite specific predictions about locality constraints in natural languages—predictions that are, as we show in our book and as we'll underscore below, confirmed.

This much said, we can turn to Fodor's particular objections. As we noted earlier, they fall into two parts: objections to our predictions about which constructions will obey Subjacency and which will not; and objections to our three key assumptions. As to the first set of objections, we'll see that while Fodor's more refined observations about what constructions obey Subjacency and what ones do not are correct, they in fact support our "leading idea" of determinism. The second set of objections center on the assumptions of determinism and its relationship to efficient parsability, our "modular" parser design and the direct embedding of grammatical representations in the parser, and the restricted space for writing down grammatical operations.

2 Determinism makes the right grammatical predictions

Turning first to the grammatical predications of our model, Fodor's interesting critique argues that our approach is both too strong and too weak. It is too strong in that our approach predicts parasitic gaps to be subject to Subjacency. This is because their deterministic detection requires scanning the left context.[1] Nonetheless, we claimed that the distribution of these categories was

[1] To show this, Fodor cites examples where in order to know whether an adjunct clause with an ambiguous verb can take a parasitic gap object, we must see whether the matrix clause contains a *wh* element in COMP. The relevant examples are contrasted in (a) and (b):

(a) What did you cook without eating?

not governed by Subjacency.

Further, our approach is too weak because it cannot distinguish a subset of gapping constructions that Fodor shows obey locality from a class that does not.[2]

First, we will show that Fodor's criticisms, while correct, deal with non-crucial assumptions of our analysis. The assumptions that replace them are fully compatible with our theory and the data cited by Fodor actually *support* our analysis in interesting ways.[3]

2.1 Parasitic gaps

The most important thing to notice about our claim that parasitic gaps are not subject to Subjacency is that it is false. Chomsky (class lectures, 1984) provides the following examples showing that these constructions are in fact subject to this constraint:

1. Who$_i$ did your read a book about e_i to e_i?

2. Which man$_i$ did you interview e_i without reading up on e_i?

*3. Which man$_i$ did you interview e_i without reading [$_{NP}$ [the file]$_j$ [$_S$ you made e_j on e_i]]?

In (1), both gaps are subjacent both from the complementizer, and from each other. This is shown by both (4) and (5), where overt movement from both the parasitic and regular gap positions is acceptable.

4. Who$_i$ did you read a book about e_i.

5. Who$_i$ did you read the book (that Mary bought yesterday) to e_i.

(b) Can you watch TV without eating?

In the second example, *eating* is unambiguously an intransitive verb, because there is no *wh* movement in the matrix clause.

[2]Before turning to these specific cases, let us dispense with one of Fodor's more general criticisms; namely, since the solution adopted does not solve all cases of parsing ambiguity, it is dubious from the evolutionary perspective. In fact, this kind of *compromise* is typical of what one finds in natural selection. The evolutionary literature abounds with cases where selection has opted for solutions that either solve part of an evolutionary problem or created other problems. (See footnote 16 of Berwick and Weinberg 1982.) Indeed Gould (1983) cautions us against adaptationists who theorized "a world of perfect design, not much different from that 'concoted' by 18th century natural theologians who 'proved' God's existence by the perfect architecture of organisms ... we do not inhabit a perfected world where natural selection ruthlessly scrutinizes all organic structures and then molds them for optimal utility." (1983:155–156).

[3]The following is a very condensed version of Weinberg (forthcoming).

Chomsky uses the contrast in (2) and (3) to argue that parasitic gaps are bound to empty operators and are licit only if they are subjacent to these operators. These empty operators are interpreted as marks of predication and so must appear at the head of the adjunct clause.[4] Put in terms of our parsing model, we can use the presence of the overt operator to signal the presence of the "real" gap. The placement of the empty operator is governed by the independent principles of $\bar{A}$ binding. The presence of the empty operator, in turn, can be used to signal the presence of the parasitic gap, if it is in a subjacent position.[5] In addition, Chomsky assumes that the theory of government interacts with the theory of bounding in that only ungoverned nodes count for bounding. Therefore, we will assume that the empty operator is subjacent to the real operator.[6] This analysis predicts that (3) is bad because, as a sign of predication between the relative clause and the head of the complex NP, the empty operator inside this relative must be bound to (coindexed with) the head. Coindexing the parasitic gap to this operator as well will result in an ill-formed structure, because quantifiers cannot be bound to two variables, as in (6). Neither the overt operator at the head of the sentence, nor the empty operator at the head of the

[4] Alternatively, following Aoun and Clark (1985), we can claim that empty operators count as $\bar{A}$ anaphors and so obey the locality conditions that apply to this class. See Weinberg (forthcoming) and Aoun, Hornstein, Lightfoot, and Weinberg (forthcoming) for details.

[5] This contrasts with Chomsky (1982) where parasitic gaps are considered underlying PROs. Brody (1983) provides independent arguments showing this account of the distribution of parasitic gaps is inadequate because it relies on the so-called *functional definition* of empty categories. In addition, the earlier analysis would obviously not predict the observed distribution of the data, since PROs are typically not bound by operators, empty or otherwise.

[6] Chomsky must argue that *all* ungoverned nodes (not just NP or S) are bounding with respect to Subjacency. This is because he wants to rule out direct movement from an adjunct as in (a):

(a) *Which article did John read a book before filing

In order to rule this out using Subjacency, he must claim that both PP and $\bar{S}$ count as bounding nodes. Moreover, he *must* use Subjacency to rule these cases out, because this is the only S-structure condition available to him and the bounding constraint in these constructions is an S-structure phenomenon, as shown by the grammaticality of (b):

(b) Who read a book before filing which article?

In Weinberg (forthcoming) and in Wahl (forthcoming) it is argued that the requirement of lexical proper government in Chomsky's ECP actually applies a the level of phonetic form (PF). This allows us to rule out a case like (a) by claimingthat the trace in the COMP of the adjunct is not properly governed, as shown in the structure (c):

(c) *[$_{\bar{S}}$ Which article$_i$ [did John read a book [before [$_{\bar{S}}$ e$_i$ [PRO filing e$_i$]]]]

Therefore, we can maintain the position that only S and NP count for the bounding system. Thus the empty operator is subjacent to the real operator in parasitic gap constructions.

adjunct are subjacent to the gap, and so they cannot license it. Therefore this structure is ruled out. This contrasts with (2), where every trace is subjacent to the operator that licenses it, as shown in (7).

*6. Which man$_i$ [S did you [VP$_j$ interview e_i][PP without [OP2$_j$ [S PRO reading [NP the file$_i$ [S' OP$_i$ [S that you made on$_j$ e_i]]]]]]]

7. Who$_i$ [S did you [VP interview e_i][PP without [S' OP$_i$ [PRO reading up on e_i]]]]?

Thus in fact, Fodor is correct in claiming that our analysis *should* predict that parasitic gaps are governed by Subjacency and we were mistaken when we claimed in our book that it did not. But we were all incorrect in believing that the constraint did not hold. Assuming that we can show that the creation of empty operators causes no problems for a deterministic system, we can use their presence to license parasitic gaps in the appropriate structures. Thus we can make the parsing model predict the properties of this construction in a straightforward and independently motivated way. It is important to note at this point that we are not changing assumptions in an ad hoc way simply to model the facts. The problem with our first attempt was that we did not follow the logic of our predictions clearly. The model *actually* predicts that parasitic gaps *should* be governed by Subjacency, as Fodor notes in her article. In the next section, we will show that the model is non-ad hoc in other ways, in that it or something like this model is needed to solve a general parsing problem that is independent of the determinism issue.

In this section, we present an algorithm to create empty operators that is also compatible with a deterministic approach. Note that the case of empty operators in adjuncts is similar to the case of factive Noun Phrases cited by Fodor in her criticism of Marcus. As in factives, the presence of the overt operator makes parasitic gaps *possible* in adjunct positions, but it does not make them obligatory in these structures. Consider (8)-(10).

8. Who did you meet without greeting.

9. Who did you meet without greeting him.

10. Who did you meet without clearing the rendezvous with security.

In a case like (8), the parser must place an empty operator in the complementizer of the adjunct phrase in order to bind the empty parasitic object of the verb *greeting*. In (9) and (10) by contrast, we do not want to place an empty operator in this position, because there is no parasitic gap in the adjunct for the operator to bind.[7] In (9) the parasitic gap is filled by a pronoun and in (1),

[7]If these operators are available at all stages of comprehension then the fact that the empty operator has no variable to bind should make the sentence as bad as (a):

there is no corresponding gap position at all. Because of the possibility of successive cyclic movement however, the parasitic gap can be indefinitely far away on the surface from the empty operator position. A deterministic parser with limited lookahead will not be able to wait for the disambiguating right context.[8] Therefore, there will be certain cases it will incorrectly place an empty operator in the adjunct's COMP.

Fodor implies that these facts pose a problem solely for deterministic parser, suggesting that a nondeterministic solution is called for. In fact, the deterministic/nondeterministic issue is beside the point. If the distinction is between a deterministic parser and a nondeterministic parser that backtracks (Fodor's choice), then both will have problems because they both at least superficially predict that such cases cause people to have noticeable difficulties in comprehending these sorts of sentences. But none of (8) (10) are difficult to understand.

The nondeterministic parsers with backtracking that Fodor cites divide cases of possible parser error into three types:

(a) *Cases that are locally ambiguous but cause the parser no difficulty.* Here it is claimed that either the backtracking needed to transform an incorrect false start into a correct analysis is so minor that it is not associated with a computational cost, or that these parsers use an exact analog of a deterministic parser's local buffer solution and thus always make the right choice. Some examples of this kind of case are given in (11).

11a. John believes Bill.

11b. John believes Bill is a fool.

Even if the parser mistakenly hypothesized that the subject of the embedded infinitival was the direct object of the verb *believe*, the backtracking needed to insert the infinitival S marker between it and verb is minor and a nondeterministic parser might be able to correct its mistake in a way that is relatively cost-free.[9]

In contrast, there are cases that require more extensive backtracking over essentially unbounded distances. These cases can be divided into two types.

(b) *Cases for which people register a strong preference for one of the possible analyses* (even when pragmatic biasing points to the other choice, but where

(a) Who did John meet Mary?

[8]The requirement that lookahead be *limited* is crucial because, as Marcus (1980) notes, a deterministic parser with unlimited lookahead could well turn out to be able to simulate a nondeterministic machine.

[9]Note that this is true even for a deterministic parser, since we need only add a new piece of information. See the next section for a related example.

both readings are eventually available). An example of this case is shown in (12), where, as Fodor mentions, there is an initial preference for the reading where *who* is taken to be the subject of an embedded clause.

12. Who$_i$ did the little girl beg to sing those stupid French songs (for) e$_i$?

(c) *Cases of conscious garden paths where one reading is difficult.* These are cases where the alternative has to be pointed out, even if it is the only reading resulting in a grammatical sentence. These include the classic sentences as in (13):

13. The horse raced past the barn fell.

The processing load here might be compatible with a backtracking approach if it is assumed that backtracking over long distances is computational costly. (It can often be difficult to assess these effects in a backtracking model; see the next section.) The extra burden imposed by true garden paths is a complex effect that is partly lexical, partly structural, and exacerbated by distance (in terms of number of alternative, but unconsidered pathways).

Cases like (8) (10) cause problems for the backtracking approach because they break the association between the extent of backtracking necessary to correct false starts and perceived sentence complexity. None of the examples in (8)–(10) produce processing complexity. This shows that there is not even a preference for adjuncts with or without parasitic gaps. Whatever the first hypothesis of the (deterministic or backtracking) parser– whether it inserts an empty operator in the adjunct's complementizer or not– one of the structures is incorrectly predicted to be difficult to process because of extensive backtracking from the site of the disambiguating parasitic gap or end of the adjunct needed to correct the mistake. (14a) and (14b) show that no extra processing complexity is observed even in cases where the disambiguating right context is very far away from the point where the decision about whether to insert an empty operator must be made.

14a. Who did you search for without telling Sue to convince Bill to ask Harry to come with you?

14b. Who did you search for without telling Bill to ask Sue to inform Harry that you would meet?

It seems then that these kind of sentences are problems for both deterministic and nondeterministic (backtracking) parsers. We could solve them if we could design an algorithm in which the semantic component simply didn't interpret empty operators unless they were eventually bound to elements in argument positions. Since these elements have no phonetic content, if they received no

semantic interpretation, it would be as if these elements never existed.[10] In that case we could insert the empty operator in all sentences, but we would be sure to be right because an unbound empty operator would simply be ignored, because it is invisible. In fact the two stage parsing model discussed in our book provides just such a mechanism.

We argued on conceptual and psycholinguistic grounds that the natural language processor was a two stage mechanism. The first stage dealt with tree expansion and the second dealt with indexation. In addition to having a different function, the second stage worked on a different representation. During the first stage, the completion of a category signaled the parser to shunt the category's daughter into a separate stack, which we called the Propositional Node Stack (PNS). The intuition behind this shunting was that once a category's thematic role was established from its position in the syntactic tree, the parser wouldn't need to retain many of the details of syntactic structure. We showed that elements in the same c-command domain are not put in the PNS until all categories in the domain are complete. This algorithm allowed the parser to correctly compute c-command relations between categories. This was crucial since these relations govern the application of the binding operations on the previously expanded tree. Pursuing the intuition that the PNS was a representation concerned with purely semantic aspects of the interpretation, we placed a semantic visibility condition on the categories appearing in this component. We claimed that to be interpreted by the semantic component (PNS), a category had to have semantic features. These were the features that allowed a Noun Phrase to either denote an individual or a set of individuals or allowed a quantifier to delimit a scope.[11] Assuming a category had such features it would be given a "referential index" and be visible in the PNS. If a category did not intrinsically have such features, it could obtain a referential index by being linked to an element that did.[12] Given the shunting procedure, an element would have to be in the same c-command domain as its antecedent in order to receive a referential index before being shunted into the PNS. If an element did not receive an index before shunting, it would become invisible and receive no interpretation. This allowed us to provide a principled explanation for the fact that grammatical conditions specifying *c-commanding* antecedents seem to

[10] An alternative would obviously be to come up with an analysis that did not posit empty operators in these and related cases. Such an account is difficult to conceive of, because we would also have to account for the subjacency effects that these constructions exhibit. By this we do not mean coming up with an alternative functional explanation for Subjacency in these cases. We mean allowing the parser (or the grammar) to distinguish those cases that are grammatical from those that do not obey the constraint.

[11] Examples of categories with intrinsic semantic features are proper names like *John*, pronouns like *him wh* phrases like *what* or *which man*.

[12] Categories that have no intrinsic semantic features and so can receive referential indices only by linking are bound anaphors like *each other* or *herself*, empty NP and *wh* traces, and certain non-*wh* quantified expressions. See Weinberg (forthcoming) for details.

apply *only* to categories with no independent referential status.[13] Chomsky (1981 and 1984 class lectures) has suggested that association with a thematic (theta) role is also a necessary condition on visibility for semantic interpretation roles. We will adopt Chomsky's suggestion and state the combined condition on visibility as follows.

> 15. (Visibility Condition) To be visible in the PNS, an element must be associated with a theta role (either by occupying a theta position or binding an element in a theta position) and must have referential features (features that either designate an individual or set of individuals or that delimit a range).

We will now show that the independently motivated shunting procedure and visibility conditions give an account of empty operators that explains why they cause no processing difficulties.

Let us reconsider sentences (8)–(10). In (8), the parser recognizes that part of the sentence is an adjunct phrase. This signals the possibility of a parasitic gap in the subsequent structure. The parser therefore inserts an empty operator in the COMP position, as shown in (16):

> 16. Who$_i$ did you meet e$_i$ without [$_{\overline{S}}$ OP$_j$ ...

If the parser subsequently finds a gap position in a subjacent domain, it can create a trace and bind the operator to it, thus associating the operator with a theta position, as in (17).

> 17. Who$_i$ did you meet e$_i$ without [OP$_j$ [$_S$ greeting e$_j$]]

Before shunting into the propositional node stack, the operator must locate an antecedent in the c-command domain with a *referential* index. If it does not find one, then neither it nor its trace will be interpreted, because even though they are associated with a theta role, they are not associated with a category that delimits a range. In this case the overt operator *who* is present in the c-command domain, so both the empty operator and the trace can receive the category's referential index (i) and so be interpreted in the PNS.

Compare this to (18). In (18) below, the parser will also detect an adjunct. It will not detect an overt operator, and so no empty operator will be created. Since there is no empty operator, no parasitic gap will be created in this structure.

> 18. Did you watch the movie without [$_{\overline{S}}$ OP$_j$ [$_S$ eating]]

[13] See Berwick and Weinberg (1984, pp. 173–182) for the conceptual arguments and Weinberg (forthcoming) and Weinberg and Garrett (forthcoming) for psycholinguistic results and additional consequences of this approach.

In cases like (9) and (10) above, the adjunct and overt operator again triggers the creation of an empty operator. Since there is no gap in the adjunct phrase, the operator is not associated with a theta role. Therefore, even though there is an overt operator to link with, the empty operator does not meet the criterion for visibility at PNS and so is not interpreted.[14] Since empty operators are not interpreted unless both conditions on visibility are met, a deterministic parser can always create these categories because they can never force it to simulate nondeterminism either by backtracking or parallelism in order to correct for past mistakes. Note that this solution will only work for *empty* operators. Lexically specified elements will receive a phonetic interpretation but no semantic interpretation, a situation that will lead to unacceptability. An empty element with no semantic features, however, is neither semantically nor phonetically interpreted and so simply plays no role in the interpretation of the sentence.[15]

The astute reader will have noted an apparent problem created by this solution. Why, one might ask, if empty categories can become invisible at later stages of interpretation, must we cue their creation to the presence of overt op-

[14] This approach will also handle empty operators in *tough* movement, topicalization, relative clauses, and the factive NPs that Fodor discusses in her criticism of Mareus. As should be obvious, since all these structures also involve predication between a phrase and a head, topic, or adjective phrase, exactly the same logic applies. See Weinberg (forthcoming) for details.

[15] Throughout this account, we have assumed, contra Chomsky, that the empty operator is subjacent ot he real operator. However, this assumption is not crucial, and remains to be verified (or falsified) by some fairly subtle empirical facts. To show this, let us assume (with Chomsky) that empty operators are not in fact subjacent to real operators. Then we must predict that the possible presence of an empty operator is queued solely by the presence of the adjunct structure. So in a case like (a),

(a) Did you catch a fish without eating?

the parser couldn't mistakenly output a structure like (b):

(b) Did you catch a fish [$_{PP}$ without [OP_j [PRO eating e_j]]]

The empty operator and parasitic gap, having no referential indices, would disappear from the semantic component's representation. However, the case features on the parasitic gap would make it visible in PF. In fact, some speakers report an initial bias towards treating *eat* as a transitive verb in these structures, and thus say that the sentence sounds unacceptable. This bias interestingly does not cross over to structures where this verb is not in an adjunct:

(c) Did you think that Harry told Mary that he expected to eat?

If these sentences reflect true biases, then an algorithm based on Chomsky's definition of Subjacency would seem more appropriate. Such an account would be fully compatible with our approach at the conceptual level. We have noted cases in our book where, in order to be specifiable using terms licensed by the grammar, the Subjacency condition is in some sense "stricter" than the parser's needs. Here we have a case where a parser whose rules are written using the grammar's predicates will sometimes make mistakes. The prediction is that people will make the same mistakes. The facts here, however, are quite subtle, and since either alternative is compatible with our approach, we leave the question of whether to place the Subjacency requirements on the empty operator open.

erators? The cases that motivated the account in the first place were those in which the local subcategorization of a verb was indeterminate. Before positing an empty element after such a verb, we claimed that we had to make sure that an actual operator was present in the previously analyzed structure. However, given our present approach, one might be tempted to argue that if a verb that can be optionally transitive turns out to be used intransitively in a given structure, the gap will simply not be associated with an operator and so become invisible in the PNS. This seems to dash the motivation for restrictions on left context, crucial for the functional motivation of Subjacency in the first place. But it is only elements with no phonetic features that can escape unacceptability if they are not semantically interpreted. Since *wh* elements have case features,[16] they will be visible in the phonological component.[17] This makes certain predictions about the applicability of Subjacency to NP movement. As noted in Lasnik and Saito (1984), all the cases where we seem to need Subjacency to rule out unacceptable NP movements are actually also ruled out *redundantly* by the Empty Category Principle. Under our approach, we *predict* that NP movement should not be governed by Subjacency, thus ruling out this redundancy, always a welcome result.[18]

Looking at the distribution of parasitic gaps from the parsing perspective allows us to supplement Chomsky's analysis in important ways. It allows us to *derive* the fact that parasitic gaps *must* be licensed at S-structure. That is, we derive as a theorem the fact that quantifiers and *wh* operators that move to COMP or some other pre-S position at LF do not create acceptable parasitic gap structures, as shown by examples (19a) and (19b).

*19a. [S You [VP [VP met who$_i$] [PP without greeting e$_i$]]]

[16]See Chomsky (1981) for justification of this assumption.

[17]See Aoun and Lightfoot (1984) for discussion.

[18]See Weinberg (forthcoming) for details. Note that the non-government of NP movement by Subjacency *reinforces* the point made in Berwick and Weinberg (1984)—namely, that Subjacency governs a natural class from the parsing perspective. The example just given shows that Subjacency only governs a *subset* of the movement constructions, the gapping examples discussed later on in this section show that Subjacency governs a *subset* of the deletion constructions. From a grammatical viewpoint, this is an entirely unnatural result.

This approach also makes sense of some preliminary results reported by Frazier (1984 Nels conference) and cited by Fodor in her article. Frazier claims that eye movement tasks suggest that subjects try to fill gaps using operators that are not subjacent to them, if the verbs governing the gap position are strongly subcategorized for direct objects. The cases are like those in (a):

a. *What$_i$ did [the girl [S who won e$_i$ receive e$_i$]

Given our approach we might claim that the gap inside the island is created on the basis of the empty operator in the COMP of the relative COMP. The fact that subjects seem to look back to the overt *wh* element is compatible with our approach if we claim that this is the result of the attempt to bind this operator (an operation *not* governed by Subjacency) to the overt operator.

*19b. [Everyone [VP [VP met someone$_i$][PP without greeting e_i]]]

We know independently that that parasitic gap constructions are not licit in the real gap occurs in Subject position.[19] In addition, if our analysis is correct, the overt operator must occur in a c-commanding COMP. As mentioned, the c-command requirement is ensured by the shunting design of the parser. If an element does not c-command a category it is not visible to it and so cannot be used to create that category as we expand the parse tree. Neither the *wh* element, nor the quantifier in (19a) or (19b) c-commands the adjuncts containing the parasitic gaps. Given the above account, there will be no binder to give referential features to the empty operator in the COMPs of these adjuncts and thus neither they nor their traces will be interpreted in the PNS. Given that the input for parsing decisions is the S-structure of the sentence, the subsequent movement of a category to a c-commanding position at a post S-structure level cannot help the parser decide how to expand the parse tree. Our parsing theory can derive both the fact that Subjacency is an S-structure property and the Subjacent government of parasitic gaps along with their licensing at S-structure– the central properties of the construction.

2.2 Gapping constructions

Fodor's next criticism deals with our analysis of gapping. She is correct in claiming that our treatment does not distinguish the subset of gapping constructions that obey bounding conditions from those that do not. As she points out, escape from bounding correlates with the appearance of an auxiliary marker in the pregap position. (20) and (21) illustrate.

20a. Mary fishes in the ocean and Harry in the sea.

*20b. Mary fishes in the ocean and I think Harry in the sea.

21a. Mary has fished in the ocean and Harry has in the sea.

21b. Mary has fished in the ocean and I think Harry has in the sea.

In our previous analysis we claimed that bounding was expected in gapping constructions because the complements of the gapped verb had to be correctly attached in the VP internal or external position. Correct attachment depends on the properties of the verb. Since an overt verb is not available to direct the parser in a gapped constituent, we predicted that deterministic attachment of these complements required a look at left context (some previous conjunct

[19] See Chomsky (1982).

containing an overt verb). Given the usual requirement of bounded access to this left context, the bound constraint on these constructions followed. Since the parser faces the same problem in both types of gapping constructions, Fodor is right in claiming that we are incorrectly led to the conclusion that the presence or absence of an auxiliary marker in the gapped constituent should not influence the application of the constraint. Therefore, in countering this argument we must show that complement attachment of PPs does not require access to left context, but that there are other properties of gapping constructions that require this access *only* in cases where no overt auxiliary precedes the gapping site. Let's start with the second point first. Consider the following examples.

22. [S I consider [S Bill [VP to be a fool]]]

23. [S I consider [S Bill [NP a fool]]]

In (22) the embedded clause is an infinitival with a VP predicate and in (23) it is a small clause with an NP predicate.[20] The head of the VP predicate in (22) can be gapped, as shown in (24).

24. [S John believes [S FRED is a FOOL] and [S' HENRY [VP [V $\emptyset$] AN IDIOT]]][21]

Fodor (1975) has shown that (24) actually involves two different deletion rules. *Main Verb Deletion* eliminates the verbal *be* form and *Tense Deletion* removes the associated tense. Cast in parsing terms, the interpretation of the second conjunct involves expanding the parse tree with both an empty tense morpheme and an empty verb. Note however that the surface string in the second conjunct is locally ambiguous and could be expanded as a gapped structure or as a small clause. If we chose the small clause alternative, the sentence would be ruled out because *believe* does not take small clause complements, as shown by (25).

*25. [S I believe [S John [NP a fool]]

The only way that we can determine the proper expansion of the second conjunct in a case like (24) is by rescanning the left conjunct. Again we have a case where a deterministic tree expansion involves left context examination.

[20] The structure of small clauses is the subject of some controversy. Chomsky (1981) following Stowell (1981) argues that embedded categories like *Bill a fool* formed sentential complements (in this case with the structure [AP [NP John] a fool]). Williams (1983) argues that these categories do not form a constituent and that they are properly analyzed as [... [NP John] [NP a fool]...]. Hornstein and Lightfoot (forthcoming) argue against Williams's analysis and in favor of a modified version of the Chomsky Stowell approach. The only point relevant to this argument, however, is that the predicates of small clauses are not VPs.

[21] We follow Fodor's convention of indicating the placement of heavy stress on a word by capitalization.

Given our usual logic, we must ensure that we will never have to look at an unbounded stretch of left context. Therefore, we predict that cases involving tense deletion should obey bounding- exactly what Fodor demonstrates. As additional evidence, consider (26a). If the parsing version of tense deletion is governed by bounding, then we predict that the small clause analysis will be the only permissible expansion of the embedded clause in the second conjunct. Since *believe* doesn't take small clauses we predict the unacceptability of the structure, in contrast with the acceptable (26b).

*26a. I think Fred is a fool and Sue believes John stupid.

26b. I think Fred is a fool and Sue believes John is stupid.

In contrast, cases that involve only main verb deletion will never create the same kind of ambiguous situations. This is because the presence of an overt auxiliary unambiguously signals that a verb phrase must follow. One never finds overt auxiliaries in small clauses. Since the parser will always be right if it expands the phrase after an overt auxiliary as an empty headed VP, it will never have to scan the left conjunct. In a case like (27) it simply uses the locally available overt auxiliary to decide about subsequent expansion of the tree.

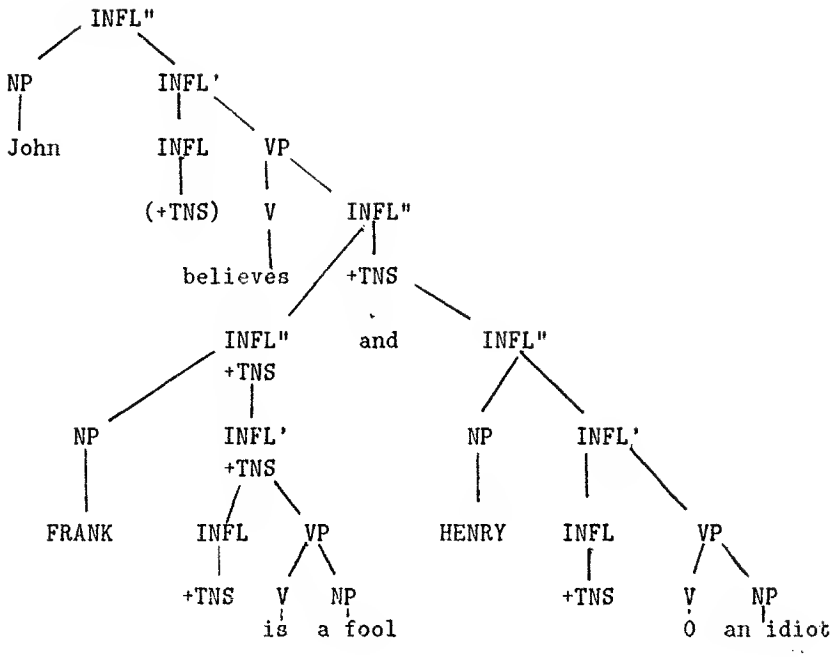
27. John has fished in the ocean and Bill has in the sea.

Since we never need to examine left context when the auxiliary remains in the surface string, we do not expect *Main Verb Deletion* to obey bounding constraints. This is in fact what Fodor observes.

This account has another virtue. The information provided by the left context to resolve the ambiguous cases will be available at the time the parser is confronted with the ambiguous material of the second conjunct. This contrasts with our previous analysis where, as Fodor correctly notes, proper identification of a verb's subcategorization and selectional properties demands access to the actual verb of the previous conjunct. Unfortunately, our parser will have already shunted this material into the PNS representation. Our parser shunts at the end of c-command domains leaving only immediate daughters of the completed constituent available as information for future parsing decisions. This is no problem for our new analysis because we distinguish small clauses from gapped constituents merely by looking at previous conjuncts for the presence of a tensed auxiliary. If we treat sentences as maximal projections of INFLection (Chomsky 1981) and if we assume that lexical information about the head of a category is projected from that head to its most maximal projection, then the relevant information will percolate up to the highest S node on the tree and thus be available to the parse for expansion decisions.[22]

[22]Projection to the most maximal projection is supported by movement of postverbal Subjects in Italian. Since these elements occur in structures like (a) we must insure that the verb

(28)



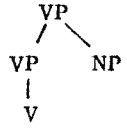
Consider again a structure like (24), repeated as (28), with irrelevant details omitted.

By the time the parser reaches the locally ambiguous second conjunct, the first conjunct will have been shunted to the PNS. Thus information contained in this conjunct will not be available for decisions about tree expansion. This causes no trouble because we see that the tensed character of the first conjunct can be read off the highest INFL projection that c-commands and is boundedly far from the INFL (INFL′) of the next conjunct. If the first conjunct was a small clause, then the 0-inflection would also percolate up to the maximal S node. This is all the information the parser needs to correctly expand the tree of the second conjunct. If the previous conjunct contains a tensed or infinitival inflection, the

---

can transmit its features to the maximal VP in order for the trace of the postverbal Subject to satisfy the conditions on proper government imposed by the ECP.

(a)



parser expands the conjunct as a gapped structure. If the previous conjunct contains a 0 inflection, then the parser expands the ambiguous structure as a small clause. This analysis makes the interesting prediction that if $\bar{S}$s instead of S's are conjoined, tense deletion should be unacceptable. Since $\bar{S}$ is not a projection of INFL, conjunction of $\bar{S}$s would not allow percolation of information beyond the first conjunct in a structure like (28).[23] Since expansion as a tensed structure is conditioned by the presence of an overt auxiliary in the previous conjunct, the parser will not be able to apply the tense deletion rule. This is confirmed by comparing (29a) and (29b), where we have conjoined S's, with (29c) and (29d), where we have conjoined $\bar{S}$s.

29a. That Frank would hit Sam and Bill would hit Harry surprised me.

29b. That [$_S$ Bill would hit Sam] and [$_S$ Frank [$_{INFL'}$ (∅) [$_{VP}$ [$_V$∅]Harry] surprised me]]

29c. That Frank would hit Sam and that Bill would hit Harry surprised me.

*29d. [$_{\bar{S}}$ [$_{\bar{S}}$ That [$_S$ Frank would hit Sam] and [$_{\bar{S}}$ that [$_S$ Bill [$_{INFL'}$∅][$_V$∅] Harry]] surprised me.]]

As predicted, Main verb deletion can apply in both conjoined S and $\bar{S}$s as shown in (30).

30a. That Frank would hit Sam and Bill would Harry surprised me.

30 b. That Frank would hit Sam and that Bill would Harry surprised me.

Thus this approach correctly distinguishes the two cases of gapping.

Returning to our first problem, we must show why the problem of complement-vs. adjunct attachment, which applies in *both* types of gapping, does not force the parser to look at left context, thus incorrectly predicting that bounding constraints apply to both kinds of gapping. The treatment in our book assumed that the semantic interpretation of adjuncts and complements proceeded in essentially the same way, by reading off tree structure. If we assume this, then it follows that a deterministic parser must attach PPs and other adjunct phrases as they are attached by the grammar, in order to carry out semantic interpretation. However, this assumption is highly dubious. As Miller and Chomksy (1963), Marcus (1980), and many others note, in certain cases, strings of adjunct phrases can occur in potentially unlimited configruations. Thus a sequence like *the man in the house by the river by the woods near the town* can have any of the following intepretations:

[23]See Zubizarretta (1982) and Stowell (1981).

[in the house][by the river [by the woods]][near the town].

[in the house [by the river][by the woods [near the town]]]

[in the house [by the river [by the woods [near the town]]]]

A parser that had to do semantic intepretation from tree structure would find itself in an exponential regress in such cases. In order to figure out which interpretation to give the sentence, it would have to compute the correct syntactic structure, but in order to do this it has to compute all the possible patterns compatible with this string, and then see which one it "means to say." This will cause an exponential slowdown in the parsing algorithm, if all trees must be explicitly reconstructed. One classic solution proposed by these authors is that adjunct phrases that can be ambiguous (either between adjunct and complement readings or between various adjunct readings) should be parsed essentially as flat structures. Semantic subroutines can then come in later and decide between the possible readings: a procedure that allows us to maintain efficient parsing.

Put in the context of the gapping constructions, if a parser cannot figure out where an adjunct is attached from the local context, it can simply attach it as a flat structure to the lowest node in the parse tree. Then, independently needed semantic routines will give this phrase its appropriate semantic interpretation. Thus the attachment of adjunct PPs in neither type of gapping can force the parser to scan left context. Therefore, the attachment of adjunct phrases does not incorrectly predict bounding effects in Main Verb Deletion.

3 Objections to basic assumptions: transparency and determinism

3.1 What is nondeterminism?

We'll first analyze the distinction between determinism and nondeterminism, and how Fodor views that distinction. Fodor makes two points:

1. A nondeterministic parser, just like a deterministic one, could benefit from locality restrictions—if the cost of backup is high.

2. A deterministic parser cannot recover from error, and so cannot comport with what is known about human processing of sentences.

Nondeterministic parsers do not reflect processing complexity

Let's take these points in turn. First, as we said earlier, one must distinguish between *two* versions of the nondeterminism hypothesis: *true* nondeterminism, where all possibilities are explored in parallel; and *simulated* nondeterminism, where one possible parse is explored at a time, and backup occurs if one line

of attack fails. Only the first version makes the nondeterministic/deterministic parsing distinction clearcut, and this is the one we chose for comparison. The second version of nondeterminism is just like the Marcus model in that a single, particular sequence of parsing decisions is made as we move through the sentence, left-to-right. It is unlike a deterministic model in that revisions in that sequence of decision are assumed to occur all the time.

Fodor does not make the clearcut choice. Instead, she opts for a deterministic, one-path-at-a-time simulation of true nondeterminism. This position is quite weak, because, as Fodor notes, one can turn this simulation into the functional equivalent of a deterministic parse simply by making the cost of revising decisions very high:

> Every point that M. makes could have been made just as well within the context of a nondeterministic parser which cared about efficiency. (Fodor, page 18)

Imposing a cost metric on backup, then, gives us more flexibility. But is this too much flexibility? There are three basic options. If we say that backup costs are zero, then we have in effect the case of true nondeterminism; if we say that backup costs are infinite, we have a Marcus model. If we make the costs somewhere in between zero and infinite, we get a middle view.

Fodor takes this as a virtue: all bases are covered. But is this so? Do we need at least this three-way split? If one is going to impose a constraint on a weaker system that has the functional effect of determinism, it would seem just as sensible to start with that constraint in the first place: assume the machine is deterministic, and see if the required psycholinguistic complexity options can be obtained this way. Cutting up the constraints this way makes a difference. A "cost" metric is the weaker position, because we must justify the metric we use somehow. That is, we must support both the assumption of nondeterminism *and* a particular cost metric. In contrast, a deterministic machine is directly built to act *as if* backtracking costs are very high. There is no separate cost metric device in the Marcus parser; therefore we need not justify one. All we need to justify is the assumption of determinism, which we must do in any case.

There could be other grounds for the flexibility allowed by a cost-metric addition to the nondeterministic model. In a footnote to her paper, Fodor tries to turn the cost-metric model to her advantage, as a way to simulate observed human sentence processing. Fodor attempts to equate backtracking cost with processing difficulty:

> But it could very well be that that the really severe garden path sentences ... are those for which all the wrong(=correct) initial choices are reconsidered before the one that was truly at fault. This is where the 2^n figure would approach a realistic estimate of parsing time, and it would nicely account for the inordinate difficulty of these sentences .... Thus the striking differences that have been observed

in the processing difficulty of natural language sentences are perfectly consistent with the mathematical results for nondeterministic parsing with online backup.

Fodor is claiming that a garden path sentence such as *the horse raced past the barn fell* demands exponential parsing time because of backup, while relatively easier "nongarden path" sentences (such as *they told the students that John liked that Bill would leave*) do not. But it is easy to see that both of these require the same amount of backtracking. The problem is that in a direct backtracking implementation, *backup occurs all the time, even on simple sentences*. For the first sentence, a backtracking parser must make a decision just before *raced*, between a relative clause and a VP. Assuming frequency preference, it takes the VP reading, which fails when *fell* is encountered. Now it must backup. We'll assume the last previous choice point was before *that John*. In fact, this is not correct. In a pure backtracking parser, we would have to unwind to all intermediate choice points: there might be a relative clause after *barn*; there might be an NP object after *raced*; and so on. Finally, we arrive at the choice at *raced* and can continue. If the machine can inspect the current word it is scanning, two or three choice points are involved.[24] More backtracking correlates with processing difficulty. Even so, such a sentence would not be *impossibly* difficult for a backtracking parser. (And remember that it would be perfectly easy for a true nondeterministic parser.) In fact, the backtracking parser does not do exponential work on such an example.

What of the second sentence? Fodor must claim that such a case causes little or no backtracking, relative to garden path sentences. But here too, a backtracking parser must do a lot of work: before *that John liked* we call for an embedded Sentence instead of a relative; similarly before *that Bill*. When we get to *would* we must backup. First, we unwind to *that Bill* and try a relative clause reading for it. This fails. Then we backup to the next previous choice point, and try alternative categorizations for *like*. Finally, we arrive at the choice between a relative and an embedded S just before *that John liked*. [25] Roughly the same backup takes place here as with the "real" garden path.

Of course, there might be some other parsing scheme to get us out of this particular dilemma. The problem is that any general scheme to make backtracking easy will almost necessarily make the garden path sentences easy as

[24] A "pure" ATN does not even look at the current word it is scanning in order to make a guess about what to do next. But this means that even very simple sentences such as *Be careful* involve extensive backtracking, because the machine guesses that it will see a declarative sentence, then a question, and so forth. This alternative would simply make our point even more strongly, so we won't adopt it.

[25] Using standard ATN techniques, preference for one type of phrase type rather than another can be encoded by ordering the arcs that leave a network state. One can order the arc alternatives so as to take a relative clause *push* after *that*, but then this will be wrong and fail to account for the preferred embedded-S reading of *they told the students that John liked the story*.

well. At heart, a backtracking parser backtracks, and it is quite difficult to use ad hoc cost measures to make it perform otherwise.

Deterministic parsers can recover from garden paths

Let's now turn to the second point, about deterministic parsing and error recovery. While Fodor wants the flexibility to simulate determinism when needed in her own model, she denies flexibility for a deterministic parser to recover from garden paths:

> The only difference between a deterministic parser and a nondeterministic parser is that in the former a garden path analysis is permanent and unrepairable, while in the latter garden paths can occur and be recovered from during the parse. (Fodor, page 18)

But again, as Fodor acknowledges in her footnote 20, this is not to deny that there could be specialized *deterministic* recovery procedures for garden path sentences, as suggested by Marcus (1980). For these procedures to apply, we would of course toe the line of determinism: backup along the lines suggested by Fodor (or in an ATN) would *not* be permitted. Ideally, following Marcus's definition, the recovery procedure should only be allowed to *add* information about the parse, not wipe out what has already been learned. Instead, when the parser blocks (because no known rule applies), a recovery procedure could look globally at the state configuration of the parser. Then, by slightly rearranging existing subtrees of the parse, the recovery procedure should simply add new information about the sentence analysis and come up with the correct sentence structure.

Interestingly enough, the Marcus design, slightly modified, provides the ingredients of just such a theory of garden path sentence recovery. We can only sketch the basic idea here.

Let us consider again *the horse raced past the barn fell*. When a Marcus-type parser fails on such a sentence, it is reading *fell*. But there is much information in its machine configuration— its pushdown stack and input buffer--of value for error recovery. It is possible to design a natural recovery procedure that uses this information deterministically to build the correct output, though at some cost. For example, in the *horse raced* example, one need only insert a new S boundary between *horse* and *raced*. There is also room within an evaluation metric of recovery to differentiate between difficult garden paths and easy-to-analyze sentences with interpretations. Barton and Berwick (1985) give some of the details. Contrary to what Fodor asserts, recovery is possible in a deterministic machine.

3.2 A two-stage design?

Fodor also takes issue with our division of parsing labor into separate tree-building and indexing stages. Again, she makes two basic points: first, that this

division is not motivated on grounds of computational efficiency; and second, that this division is not motivated by the grammar (so that we are violating our own assumption of transparency connecting grammar and parser). Again, we disagree.

Consider computational efficiency. Fodor first claims that computational reasons alone can't motivate the bounded-context character of our parser:

> Given that the efficiency results for bounded context-parsing are no better than for LR(k) parsing in general, the crucial assumption that the first stage of B&W's parser is a bounded context device receives no support from these efficiency results. (Fodor, page 41).

But as Fodor herself notes, computational complexity calculations are often relative to representational issues. If one picked some *other* representational format, then certain computational issues can become irrelevant. For example, if we adopt true nondeterminism, then it is not difficult to parse *any* sentence of a context-free grammar, no matter how ambiguous, in time proportional to the square of the grammar size and the cube of sentence length (where the grammar is measured in terms of the total number of grammatical *symbols*, like NP and VP, not just *rules*. See Earley (1968)).

This being so, one cannot divorce a discussion about computational efficiency from representational format. We have chosen to represent the parser's knowledge *transparently*, that is, to include only those categories sanctioned by the grammar. The categories of our grammar include only the basic lexical projections NP, VP, PP, and so on.[26] By saying that our parser works transparently, we mean that the parser's rules can only make reference to these literal symbols. To put the same point another way, transparency requires that the only states the parser has are the "states"—i.e., the nonterminal names—that the grammar has. The parser cannot use any *derived* facts about the grammar; nor can it appeal to nonterminal symbols that do not otherwise exist. For example, the parser cannot create a new state in order to "remember" that a *wh* phrase has been encountered earlier in the sentence. This would correspond to a complex nonterminal name such as WH/NP.

In general, LR(k) parsers are allowed to create such states whenever they are needed. These states (in the form of a finite-state control table) encode the set of possible left-most derivation patterns for the given grammar. Since they represent derivation regularities, these states need not map in a 1-1 fashion to the nonterminal names of the grammar, and in fact the wh sentence example shows that in some grammars the nonterminals do not match the states of the

[26]Like most syntactic theories since *Aspects of the Theory of Syntax*, we also include traditional agreement features like Person, Number, and Gender, as properties of lexical projections. We explicitly *do not* include the "slash" feature of Generalized Phrase Structure Grammar (resulting in complex categories like VP/NP), since this feature is not lexically projected (X^0 or lexical items are specifically barred from having "slash" features in GPSG).

parsing machine.[27] However, we have specifically barred the use of parsing states that do not correspond to lexically projected nonterminal names. Therefore, our approach does not admit the entire class of LR(k) parsers. Instead, our parsing rules can make reference only to grammatical symbols. There is a class of deterministic parsers that defines such a class of machines, namely, the *bounded-context parsers*.[28] This is the parsing design we have adopted.

Fodor is correct that general computational grounds do not force the bounded-context choice on us--but that is trivially so. For example, if we adopted a more powerful device, such as a nondeterministic device, we would not need this structure. But, all other things being equal, it is the stronger assumption. Transparency is stronger, because we need not posit any entities beyond those the grammar already gives us; and all other things are equal, because in this case "all other things" is simply parsing efficiency and an account of the psychological facts about parsing unbounded dependencies.[29] It is of course true that a parser need not respect the representations provided by the grammar. But it is simpler to assume that it does. A grammar that contains just projections of lexical items is smaller, simpler, and hence easier to learn than one that does not. There's a sense in which such a parser is completely lexically based--there are just projections of lexical items, and nothing more.

Fodor also argues that transparency itself does not motivate a literal bounded-context parser, because the grammar contains rules that mention variables: "as long as the transformational rules of the competence grammar can contain variables (explicit or implicit) we would expect parsing rules employing the same metalinguistic vocabulary to do the same." She concludes that we need "an explicit prohibition against variables in the parsing rules." (Fodor, page 47). But again, there are two parts to any computational operation: the procedure itself, and the data structure or representation it works on. In this case, there are no variables because there are no complex category symbols, and because the rules of the machine are finite. As Fodor notes, these are indeed "stipulations" (page 48)--one must always assume something in arguments about computational matters, since we don't have the luxury of neurophysiological findings.

---

[27] This transparency distinction also shows up in the way that LR(k) parsers are built. The usual approach is to process an LR(k) grammar to derive a finite-state control table that is actually used for parsing. The states of this table need not, and usually do not, correspond in any transparent way to individual nonterminal names. Instead, in effect they stand for theorems *about* derivations in a particular grammar. By banning such nontransparency, we are banning such preprocessing.

[28] See Floyd (1964). Actually, we must define an extension of the bounded-context parsers that uses nonterminal lookahead as the Marcus machine does. For details, see Berwick (1985). We could also vary other details of the bounded-context design, as long as we retain the key feature: parsing rules must refer only to grammatical symbols, not to parsing states.

[29] To make the same point in reverse, the only evidence for the more powerful machinery of a hold cell or "slashed" categories seems to be the ability to parse unbounded dependencies. But if this can be explained without resort to such machinery, then this leaves its justification unestablished.

Similarly, Fodor "stipulates" that a grammar allows machinery beyond basic $\overline{X}$ categories, and that the parser includes backtracking as a standard feature. The question is how natural these stipulations are. In fact, in Government-Binding theory, the rule Move α does not have variables (Chomsky 1977, 1981 is quite explicit on this point). Deletions, on the other hand, can have variables, but this is not relevant for parsing because deletions are locally unambiguous (see the previous section on Gapping and Berwick and Weinberg (1984)).

Beyond this question of bounded-context parsing, Fodor then goes on to question our division of parsing into two stages at all. She again claims that we violate our own criterion of transparency and that such a division is not needed on grounds of efficiency.

The efficiency counterargument, at least in one form that Fodor gives, goes something like this. Our second stage procedure that computes referential dependencies—that *John* and *he* may denote the same person in sentences like this:

John$_i$ believes that Fred thinks that Sue said that he$_i$ is smart.

Since this procedure, whatever it is, must be able to search unbounded domains, why not just let it do the job of searching for the antecedent of a *wh* phrase? Alternatively, why not just fold the two stages together, combining both jobs into one? In effect, Fodor wants to "multiply out" the two representational levels we have distinguished into a single one because this is more efficient.[30]

Since Fodor elsewhere (Crain and Fodor 1984) has herself argued for the computational *benefits* of nonmodular representations, it is worthwhile to see just what is at stake here. Fodor's support for nonmodularity is surprising. First of all, from the standpoint of computer science generally, it cuts against the grain of all that is known about the efficient solution of complex problems. (See, e.g., standard works on algorithms, such as Knuth, 1973; Aho, Hopcroft and Ullman, 1974.) Second, the key point is that for modularity to work the distinct levels should have different representational properties, because each is designed to highlight different aspects of the same problem. This is the source of the power behind the idea of two levels of representation, words and phrases. It is easier to state the facts about agreement if we use Noun Phrases and Verb Phrases rather than simple words, because then we have just two simple representational units adjacent to one another (NP next to VP). In fact, a simple finite-state automaton suffices, given that the phrases are constructed first. Similarly, there are facts about language that are more easily stated in terms of a linear arrangement of words—e.g., that a Determiner precedes a Head Noun, and may agree with it. This (oversimplified) factored representation

[30] At times, Fodor suggests just the opposite, as when she proposes that the first and second states ought to divide computational labor between them: "the first stage device might call on the second-stage device to do the antecedent check prior to trace postulation. This might call for a slightly more complicated routine to pass control back and forth between the two, but the labor saved could very well compensate." (Fodor, page 43)

can be modeled as a cascade of finite-state *transducers*, where the first level system, that of words, builds a phrasal representation and feeds the second level. Is it possible to collapse these two levels into one? Yes: one can "multiply out" all combinations of words and eliminate the phrasal level, by forming the product of the two finite-state machines representing each level (see Berwick 1982). However, it does not make sense to collapse these two levels into one. The collapsed representation is much larger, because all possible combinations of constraints, previously independently expressed at each level, are now written out explicitly. The resulting system is much larger. In general, if the constraints on one level can be expressed by a machine of size n, and the constraints on a second level can be expressed by a machine of size m, then the collapsed machine could be of size nm.[31] In fact, this is one traditional argument for a multiple-levels view of language, as initially expressed in Chomsky's *Logical Structure of Linguistic Theory*. There are two computational advantages to the modular view: one, just mentioned, is that the resulting system is easier to learn, if we equate smaller size with easier learning; the second is that we can design computational procedures tailored to work with the specific formats of each level.

This is exactly what we aimed for in our two-stage model. Each level has a different representation that highlights different aspects of the computation of linguistic structure, and each is designed to ease the computation of properties relevant to that level. The first level deals with questions of how to build a tree, and uses notions like *dominate*, *precede*. For example, in the sentence example we gave just above we expand the tree in exactly the same way no matter whether *he* is bound to *Fred* or whether it is a free pronoun bound to a discourse NP that occurred much earlier. This contrasts with cases governed by Subjacency. The presence or absence of an antecedent tells us *how* to expand the tree we are building. If there is an antecedent in the structure and a verb that selects or subcategorizes for an NP, we create a trace slot in the phrase structure; otherwise, we do not. This is a decision about tree structure.

Roughly speaking, referential dependencies can cut across sentences and involve all the objects mentioned in a discourse—plainly outside the purview of sentence tree predicates. Secondly, referential dependencies are calculated on a different representational base from phrase structure, just as Subject–Verb agreement is calculated at the level of phrases rather than words.

What would happen if we tried to collapse the referential dependency calculation together with tree-building is exactly what would happen if we tried to compute Subject-Verb agreement at the level of words. As we show in our book (Berwick and Weinberg 1984), our first stage procedure works in linear time, in time cn, where c is a constant depending on the size of the output phrasal structure and the size of the grammar, and n the length of input sentences.

[31] For more realistic representational formats, e.g., context-free grammars, the savings can be even larger. See Berwick 1982 for details. See the next section for additional comments on this problem and grammar size.

The search for referential antecedents would now have to look at a representation defined over complex tree shapes, including many irrelevant structures. We note in our book that in the worst case this would increase analysis time to kn^2, where n is the length of the input sentence, and k is some constant that depends on the size of the phrase description. It is already apparent that pronoun referential dependency can extend across sentences. It is also apparent that this computation can be nonlinear: consider the laborious calculation that seems to occur when one uses a pronoun whose antecedent lies many sentences behind in a discourse. What Fodor wants to do by combining these two steps is make the first stage procedure nonlinear as well. But as she herself notes (page 68: "in general, linear time parsing is surely just what a model of the human sentence processing mechanism should aim for"), this would have the unfortunate effect of making the construction of tree structure for single sentences potentially nonlinear. We want to avoid this. We would like to recover the right tree structure in linear time, even if the pronoun antecedents are not in place. Note that there is much we can interpret about a sentence if we have its correct phrase structure, even if we do not know that *he* is dependent on an earlier NP. Fodor's collapsed scheme in effect forces the machine to stop and wait for the right antecedent calculations to complete before plunging on.[32]

By factoring apart the stages of tree-construction and referential dependency calculation, we gain at the second stage as well because the size of the structures the search procedure works over can be made smaller. That is, instead of running our procedure in time cn^2, where c is large, we can run it in time kn^2, where k is a short list of NPs. As we noted in our book, this is a difficult argument to make because in most cases sentences are short. But let us see what it means in detail. The second-stage representation includes shunted predicates and NPs. It is a simple matter to take this propositional representation and build a finite-state transducer (standing for a homomorphism) that projects just the NPs from this second list. We may imagine this projected bag of NPs to be the discourse NPs for this sentence; it could include, perhaps, the NPs for previous sentences—but just NPs. It is because we have now isolated these units on a separate level that the search for referential dependents is easier. No other units stand in the way of a direct search through the NP list. In most cases, there will be only a few NPs to look at. Note that this method only works because we have set up the first stage to build just the right structured list so as to provide the right NPs to look through. Further, in those cases where the list is large, we expect to find nonlinear processing difficulty – informally at least, precisely what seems to happen when there are many potential NP antecedents.[33]

[32]One could design a "pipelined" scheme where a second-stage referential dependency calculation works off the input from a first-stage device. But this is just our two-stage model in another guise.

[33]That is, a linear list of this kind, if long enough and if it included discourse NPs, might take linear time to search for any single NP. Of course, there are other possibilities, since

To summarize, we argue that isolating the referential dependency calculation in this way pinpoints an important functional distinction between building tree structure and referential dependency. Tree construction is fast (linear time, and, in fact, realtime if one examines our procedure in detail): each phrase is built in a bounded amount of time; coindexing (or referential dependency calculation) does not interfere with this, for it can be nonlinear. Fodor's proposed one-stage model, because it interweaves these functionally distinct processes, slows both down.

3.3 Another source for locality principles?

Finally, Fodor contends that locality principles could be motivated in a GPSG-type theory, both on grounds of easy parsability, and—another point that we ourselves note—on grounds of learnability:

> This negative result does not mean that subjacency could not be functionally grounded in a GPSG. As chapter 3 observed, there are many possible "functional" constraints that could have played a role in the shaping of language. Foremost among these, at least traditionally, is learnability. (Berwick and Weinberg 1984:166)

Fodor makes two specific proposals along these lines, one for parsability, and one for parsability/learnability. Let's take each in turn.

Consider first her argument that a GPSG parser would benefit from locality constraints resolved by context on the *right*, in sentences such as *Who did you help* ..., where the parser must decide whether to insert a trace after *help* or keep going so that the trace will appear in some lower complement. But once again, this constraint just doesn't matter under the true nondeterministic model. Advocates of GPSG often cite the parsing results for general context-free grammars as evidence that such a system will work efficiently. But then, Fodor's demand for constraints on context become more mysterious. Suppose one uses Earley's parser for context-free grammars. This is one standard algorithm on which the efficiency results for generalized phrase structure grammar are often based. Then all parses are kept in parallel, and there's no problem at all: both alternatives are carried along, and when the problematic gap appears or fails to appear, one of the possibilities falls by the wayside. There is no reason that the locality constraint must exist. The point is not that the GPSG parser *cannot* be made to benefit from a locality constraint but that it *doesn't need* to benefit from a locality constraint in the right-context situation.[34]

not much is known about the representation of semantic structures. For example, it could be that such NPs can be accessed in constant time, up to a certain memory limit—as if one could instantly remember the last 10 things mentioned. If so, then processing difficulties might not show up on short sentences. Like so many other details about processing, this one hinges on representational questions that we cannot answer in detail as yet.

[34] Alternatively, one could dispense with the Earley algorithm and come up with some other parsing algorithm for these systems. But then it remains to establish that this alternative

What about our trace-based parser, then? Why can't we add similar parallelism and thus avoid the need for a locality constraint? Remember that our parser design does not have complex categories such as S/NP, VP/NP, and so on; it can use just the unalloyed categories provided by $\overline{X}$ theory. It does not use a hold cell, or any other special memory. Given these transparency constraints, it is interesting that while true nondeterminism will make a locality constraint for right-disambiguating contexts superfluous, it actually leaves the demand for Subjacency unscathed. Consider what happens if we had a true nondeterministic, trace-based analysis of sentences such as, *What did Mary say ... that John ate?*. Note that the analysis is completely determined *up to the point that the "gap" after eat is encountered.* That is, the parser is not carrying along two analyses at this point, as it is in the right-context case. At *ate* the parser takes the nondeterministic solution: it writes out one parse with the trace inserted, and one with it not inserted. But now what? The sentence ends. No additional information is forthcoming, and yet there are still two viable analyses of the sentence. One of these is grammatical (where the trace is inserted) and the other is not. ambiguous. But the sentence is not interpreted as having two analyses, one grammatical, one not. There is no evident way to force the other reading out. Thus, the nondeterministic analysis actually makes things worse here: it yields two candidate interpretations when only one will suffice. To resolve these, we must now rescan the output analysis tree, to pick up whether a *wh* was present—adding to the computational cost. Right-context won't help us here, because there is no right-context. But there's no evidence that this reanalysis occurs, or that such a sentence is hard to process. We conclude that nondeterminism does not help us if we have only the categories S, NP, VP, etc. and no Subjacency ; on the contrary, it hurts. Thus, Subjacency is still predicted in our model, unlike Fodor's. Note that this is quite unlike the right-disambiguating context case, where pursuing alternatives in parallel allowed us to hold off making a decision until information became available.

What about the second proposal, about learning? Just before her conclusion, Fodor suggests that a GPSG system might need locality constraints to make its rule system smaller, hence more easily parsable, and, as suggested in the other papers where she has advanced this proposal (Fodor 1984) more learnable.

In the absence of any details about just how easy or hard it is to parse a full-scale derived rule system, it is difficult to judge this proposal. We must first emphasize that Fodor here is talking about a grammar that explicitly lists possible phrase structure patterns rule by rule. This is rather different from the current GPSG framework that represents a grammar via a set of dominance and precedence statements (ID/LP format) for basic phrasal relationships, implicational statements to encode feature redundancies, and metarules to account for systematicities like active-passive sentences (Gazdar, Klein, Pullum, and Sag, 1985). What one finds is that in any reasonably full-scale grammar, for, say,

parsing method—whatever it is—is efficient. Fodor does not offer a concrete alternative.

English, the explicit rule system is so large that there's only marginal gain in "reducing" the size of an explicit rule system in the manner Fodor suggests. This is because the reduction is miniscule compared to the total overall size of the rule systems themselves. Let's see why this is so.

To begin, we must be precise. Since Fodor wants to make an argument about improving parsing efficiency by reducing grammar size, let us define grammar size, $|G|$, as the *total number of symbols in the grammar accessed for parsing*. This is the standard measure. (See Earley 1968 for discussion.) We do not want to use the total number of individual *rules* of the grammar, because this would weight against rule systems with "short" rules (e.g., A→BC; B→DEF as opposed to A→DEFC).

Let us now compare the grammar size of an explicit phrase structure rule system that allows a one-S extraction constraint vs. one that allows extraction across three S's. Elsewhere (Fodor 1984), Fodor has suggested this as an example of the benefits of constraints: the tighter the constraints on extraction, the fewer the rules. While this is literally true, the problem is that such a grammar is already so large that any minor effect imposed by one new constraint is swamped out.

It is of course quite difficult to know what the "true" grammar size for such a system is, because we do not know what the "true" grammar of any natural language is, even of English. However, we can say this much: any such explicit rule system must have a rule for every possible surface phrase structure pattern. How many such patterns are there? Perhaps the most systematic study of such patterns has been carried out in the context of Sager's work (1981). For instance, Hobbs (1974) estimates that a subpart of the Sager grammar, when expanded out into a context-free form, would be "about several orders of magnitude larger" than the 200 productions and 300 context restrictions it contains in context-sensitive form (1974:132). That is, the expanded grammar size would be have about 20,000 60,000 context-free rules .[35] We take this as a fairly conservative estimate of the number of explicit, rule-by-rule descriptions of phrase structure patterns in English.[36]

The Earley algorithm runs in time at most $|G|^2 n^3$, where n is the sentence length in tokens. That is, using the Earley algorithm with a fully-expanded,

[35] The initial grammar's productions are in Chomsky normal form, and therefore have a size of 3 per production. Thus the initial grammar size is about 600, with 300 context restrictions.

[36] Note that most grammatical descriptions that appear in the computational literature in fact describe only small fragments of natural languages—quite reasonably, since they are often designed to illustrate one or another theoretical point, or work within a sublanguage that serves some functional end (like database retrieval); they are not designed for broad coverage. For instance, the example GPSG system described by Gawron, King, Lamping, Loebner, Paulson, Pullum, Sag, and Wasow, 1982 for database retrieval has an expanded grammar size of about 1500 1800 (1982:77), but does not include many sentence types and restrictions of the Sager grammar. For instance, appositives and sentence adjuncts of many different types are not included (*Little did she know that . . .*; *Whatever you say, the guy, the very same person you saw yesterday, is . . .*).

explicit rule system for English, the running time would be at worst $1.6 \times 10^9 n^3$, or about a billion $\times n^3$. The result is that any change brought about by introducing a constraint on extraction across one S rather than, say, three, is irrelevant. The base grammar with three-S extraction will need two or three extra nonterminal symbols, in order to "count" how many S's have been crossed (S_1, S_2, S_3). Suppose this adds 50 new rules. What happens to parsing time? It is "exploded" from 1.5 billion n^3 to 2.4 billion n^3—an increase, to be sure, but one that cannot possibly matter, because the constant factor is already so large.

We do not mean to take this as a serious calculation; it is quite speculative. However, the qualititative point still stands. This exercise is simply designed to demonstrate that an explicit rule system doesn't exhibit the right kind of demarcation between *one* and *more than one* that is so characteristic of natural languages. Details about grammar size aside, if extraction across two domains does not lead to a processing burden, then it is hard to say why three rather than four or five domains does. Any system grounded on explicit phrase structure rules does not naturally distinguish between a locality condition that acts over, say, *three* domains and one that acts over a single domain. We just saw that there could be no relevant difference for parsing, or for learning (if we equate size of rule system with difficulty of learning). But we suspect that this simply misses an important property of natural grammars: namely, that they do not have "counting" predicates that distinguish between two or three, or 17 domains. This is evidently a property of grammars generally, and has some power in explaining the metrical structure of phonological rule systems (see Halle and Vergnaud forthcoming 1985). But *why* do grammars have this property? If we assume that rule systems are written in a derived fashion, as Fodor insists, then there is no reason for it. A grammar that counts to 16 is just as easily parsed and just as easily learned as one that does not.

Suppose, in contrast, that there are no phrase structure rules—no explicit derived rules at all. Instead, suppose that there are just individual lexical items and their feature projections (as defined by $\overline{X}$ theory), plus the movement rules and constraints defined by GB theory. Now there cannot be any rule of grammar that cuts across just *three* S domains. Individual lexical items can subcategorize for single S's, and hence build phrases consisting of *adjacent* S domains. Since movement can apply, we can move elements across these domains. Cyclicity (iteration of this process) leads to superficially unbounded movement. *But no other constraints can even be stated.* The vocabulary for writing down grammars cannot refer to phrase structure rules, and so cannot write down a chain of three S expansions to allow extraction across three S's but not four. As we observed in our book, *either* free (unbounded) movement is possible, or else movement across a single category is blocked; nothing in between is allowed. This result – the noncounting evidently true of natural grammars—*follows* from

the nonexistence of derived phrase structure rules.[37]

Of course, nondeterminism and the flexibility allowed in writing derived grammars leaves open many possibilities. As we have seen, this is exactly what is wrong with a weak set of hypotheses: it leaves open too many avenues to explore. As we said at the outset, we prefer to tackle the problem head on, by adopting strong constraints that lead to interesting predictions and explanations of why natural grammars are built the way they are, giving up those constraints only when absolutely necessary. So far, we've been encouraged by the results. Our predictions about locality principles, suitably revised, hold up. Our modular design leads to testable hypotheses about the role of c-command in language processing, now being probed (Weinberg and Garrett, forthcoming). Our transparency assumption leads to noncounting grammars. We see no reason to abandon the chase now, when we have come so far.

[37]As far as we can tell, this property also holds in current GPSG frameworks that avoid explicit phrase structure rules and use subcategorization and ID/LP statements instead to define a set of admissible phrase structures. Thus this version of GPSG also obeys noncounting.

References

Aho, A., J. Hopcroft and J. Ullman, 1974. *The Design and Analysis of Computer Algorithms*, Reading, MA: Addison-Wesley.

Aoun, J. and R. Clarke, 1985. On empty operators. University of Southern California Working Papers.

Aoun, J., N. Hornstein, D. Lightfoot, and A. Weinberg, forthcoming. Two notions of locality. University of Maryland College Park and University of Southern California, unpublished ms.

Aoun, J. and D. Lightfoot, 1984. Government and contraction. *Linguistic Inquiry*, 15, 465-473.

Berwick, R., 1982. Locality Principles and the Acquisition of Syntactic Knowledge. PhD disseration, MIT Department of Computer Science and Electrical Engineering.

Berwick, R., 1985. *The Acquisition of Syntactic Knowledge*. Cambridge, MA: MIT Press.

Berwick, R., and E. Barton, 1985. Assertion set parsing. Artificial Intelligence Laboratory, MIT, ms.

Berwick, R., and A. Weinberg, 1982. Parsing efficiency, computational complexity, and the evaluation of grammatical theories. *Linguistic Inquiry*, 13:165-191.

Berwick, R. and A. Weinberg, 1984. *The Grammatical Basis of Linguistic Performance*. Cambridge, MA: MIT Press.

Brody, M., 1983. On contextual definitions and the role of chains. *Linguistic Inquiry*, 15, 355-380.

Chomsky, N., 1977. On wh-movement. In *Formal Syntax*, P. Culicover, T. Wasow, A. Akmajian, eds., pp. 71-132.

Chomsky, N., 1981. *Lectures on Government and Binding*. Dordrecht: Foris Publications.

Chomsky, N., 1982. *Some Concepts and Consequences of the Theory of Government and Binding*, Cambridge, MA: MIT Press.

Crain, S. and J. Fodor, 1984. How can grammars help parsers? in D. Dowty, L. Kartunnen, and A. Zwicky, eds., *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, Cambridge, England: Cambridge University Press.

Earley, J., 1968. An efficient context-free parsing algorithm. PhD dissertation, Carnegie-Mellon University Dept. of Computer Science.

Floyd, R., 1964. Bounded context syntactic analysis. *Communications of the Association for Computing Machinery*, 7:2, 62–67.

Fodor, J.D., 1975. Gapping gapped. Unpublished ms, University of Connecticut.

Fodor, J.D., 1984. Learnability and parsability: a reply to Culicover. *Natural Language and Linguistic Theory*, 2, 105–150.

Gawron, J.M., King, J., Lamping, J., Loebner, E., Paulson, A., Pullum, G., Sag, I., and Wasow, T., 1985. Processing English with a Generalized Phrase Structure Grammar. *Proc. of the 20th Annual Meeting of the Association for Computational Linguistics*, 74 81, Toronto, Canada.

Gazdar, G., Klein, E., Pullum, G., and Sag, I., 1985. *Generalized Phrase Structure Grammar*, London: Basil Blackwell.

Gould, S., 1983. *Hens Teeth and Horses Toes* New York: Norton.

Halle, M. and J. Vergnaud, 1985 forthcoming. *The Metrical Theory of Phonology*, in press.

Hobbs, J., 1974. *A Metalanguage for expressing grammatical restrictions in nodal spans parsing of natural language*. Courant Institute of Mathematical Sciences, Report NSO-2.

Hornstein, N. and D. Lightfoot, forthcoming. On predication. University of Maryland College Park.

Knuth, D., 1973. *The Art of Computer Programming*, Reading, MA: Addison-Wesley.

Lasnik, H. and M. Saito, 1984. On the nature of proper government. *Linguistic Inquiry*, 15, pp. 235–290.

Marcus, M., 1980. *A Theory of Syntactic Recognition for Natural Language*, Cambridge, MA: MIT Press.

Miller, G., and N. Chomsky, 1963. Finitary models of language users. R. Luce, R. Bush, and E. Galanter, eds., *Handbook of Mathematical Psychology*, NY: John Wiley, 419-491.

Sager, N., 1981. *Natural Language Information Processing*, Reading, MA: Addison-Wesley.

Stowell, T., 1981. Origins of phrase structure. PhD dissertation, MIT Department of Linguistics and Philosophy.

Wahl, A., forthcoming. Two notions of locality. unpublished ms., University of Maryland, College Park, and University of Southern California.

Weinberg, A., forthcoming. Topics in parsing and GB theory. PhD dissertation, MIT Department of Linguistics and Philosophy.

Weinberg, A., and Garrett, M., forthcoming. Parsing and c-command. Ms., MIT Department of Psychology and University of Maryland, College Park, Department of Linguistics.

Williams, E., 1983. Against small clauses. *Linguistic Inquiry*, 14, 287–308.

Zubizaretta, M., 1982. On the relation of the lexicon to syntax. PhD dissertation, MIT Department of Linguistics and Philosophy.